REMARKS

INTRODUCTION

Claims 1-26 were previously and are currently pending and under consideration.

Claims 1-26 are rejected.

Claims 1, 3, 5, 7-10, 12, 14-20, 23, 24, and 26 are amended herein.

No new matter is being presented, and approval and entry are respectfully requested.

REJECTIONS UNDER 35 USC § 102

In the Office Action, at pages 2 and 3, claims 1, 3, 10, 12, 16, 17, 20, and 21 were rejected under 35 U.S.C. § 102 as anticipated by Bereiter. This rejection is traversed and reconsideration is requested.

BEREITER DOES NOT COLLECT NONVOLATILE INSTALLATIONS WITHOUT REGARD FOR EXECUTION

Claim 1, for example, recites "collecting inventory information from each of a plurality of computers within the organization, the inventory information including information regarding software actually installed nonvolatiley in the computers, where the inventory information is collected without regard for whether or not the installed software is currently executing".

The Microsoft Computer Dictionary (4th ed., copy included) states that "install" can indicate "[t]o set in place and prepare for operation. Operating systems and application programs commonly include a desk-based installation, or set up, program that does most of the work of preparing the program to work with the computer, printer, and other devices. Often such a program can check for devices attached to the system, request the user to choose from sets of options, create a place for the program on the hard disk, and modify system startup files as necessary." The amended claims recited features of "installed" software such as being nonvolatile ("retaining data when power is shut off", Merriam Webster Dictionary), and software is "installed" whether or not it is actually executing. Often, the very purpose of installing software is to provide the capacity stop and start the installed software. In sum, although "installed" is qualified in the claims as "nonvolatile", "installed" is also a term of art with specific meaning by itself, and that meaning differs significantly from the meaning of "executing", "invoked", etc.

According to its abstract, Bereiter counts a number of simultaneous invocations of one or more application programs that occur in response to execution of corresponding system management tasks or processes. In other words, Bereiter counts a number of executions of applications or a number of the executing applications. A number of installations is not correlative with a number of executions. An execution or invocation can occur on a computer without necessarily requiring an installation (e.g. Application Server Programs or ASPs, Cytrix Windows Application Servers, Java Applets executing on clients but not installed thereon, etc.) Also, nonvolatiley installed software can be installed without necessarily being executed.

Therefore, it is respectfully submitted that Bereiter differs in that it collects execution information which is significantly different than installation information. Bereiter cannot be said to collect licensing information "without regard for whether or not ... installed software is currently executing". See Figure 7 of Bereiter, which shows counting 104 the number of invocations not installations. See also the Abstract, which mentions "deriving a count of a number of ... invocations" ("invoke ... to put into effect or operation").

Applicant respectfully notes that the "in use" recited in claim 1, for example, refers to the license; it is the license that is "in use", not the installed software "in use" by way of execution. The "in use" is based on the collected installation information, which is collected regardless of execution.

Withdrawal of the rejection of independent claims 1, 3, 10, 12, 16, 17, and 20 is respectfully requested.

REJECTIONS UNDER 35 USC § 103

In the Office Action, at pages 4-8, claims 2, 4-9, 11, 13-15, 18, 19, and 23-26 were rejected under 35 U.S.C. § 103 as obvious over Bereiter in view of Duvvoori. This rejection is traversed and reconsideration is requested.

The independent claims are distinguishable for reasons discussed above with reference to Bereiter. Duvvoori was not cited for and does not disclose or suggest counting software installations. Duvvoori, like Bereiter, concerns "the number of copies of licensed programs in execution at any particular time" (Abstract).

Claims 7, 14, and 19 recite distributing a dictionary for the purpose of counting software installations. As discussed above, the prior art combination does not count software

applications. Therefore, the proposed prior art dictionary does not count software installations.

Claims 8 and 15 recite features related to purchasing software licenses when it has been automatically determined that additional licenses are necessary. In particular, claims 8 and 15 recite "in a case that the sum of the software-license usage numbers of all sections of the organization exceeds the software-license holding number, in response to such a determination automatically generating a purchase transaction for purchasing software licenses based on a difference between the sum and the software-license holding number".

In Bereiter, the remedial action is taken by an administrator or other person. There is no discussion of responding to an automated license deficiency determination by automatically generating a purchase. None of the cited references share the inventors' observation that licensing violations can be proactively prevented by connecting automated license deficiency determination to automated purchasing. As the references note, a copyright violation can cause severe penalties. Claims 8 and 15 provide a synergistic benefit of helping a company avoid penalties. Bereiter's administrators and prior art manual purchasing processes are slow and prone to mistakes. Withdrawal of the rejection is respectfully requested.

DEPENDENT CLAIMS

The dependent claims are deemed patentable due at least to their dependence from allowable independent claims. These claims are also patentable due to their recitation of independently distinguishing features. For example, claim 22 recites "a hierarchy of the organization displayed in the form of a tree and the state of usage of software licenses within each selected section within the hierarchy". This feature is not taught or suggested by the prior art.

Under the rejection's reasoning, any new use of a visualization hierarchy must be a priori obvious because it would improve viewing. The motive for modifying claim 22 is too general to support a prima facie case of obviousness. Although the proposed advantage might occur after the modification has been made, the rejection must indicate where the prior art specifically suggests, for example, that it is desirable to improve visualization of an organization's licensing information, that there is a problem in this area, etc. Almost anything, at a basic enough level, is well known. For example, a resistor and its advantages is well known, however, a resistor can have unobvious new uses not specifically suggested by the prior art. Because an old

component has inherent advantages does not mean that it is obvious to apply it in all situations. Withdrawal of the rejection of the dependent claims is respectfully requested.

MISCELLANEOUS CHANGES

Other claim changes not discussed above have been made to broaden the scope of the claims or for further clarity.

CONCLUSION

There being no further outstanding objections or rejections, it is submitted that the application is in condition for allowance. An early action to that effect is courteously solicited.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 8 April 2004

By: James Strom
James T. Strom
Registration No. 48,702

700 Eleventh Street, NW, Suite 500
Washington, D.C. 20001
(202) 434-1500

CERTIFICATE UNDER 37 CFR 1.8(a)
I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
on April 8, 2004
STAAS & HALSEY
By: James Strom
Date: 8 April 2004

Designed for Microsoft® Windows NT® Windows® 98

CD-ROM Included

Microsoft® Internet Explorer

THE ULTIMATE COMPUTER REFERENCE

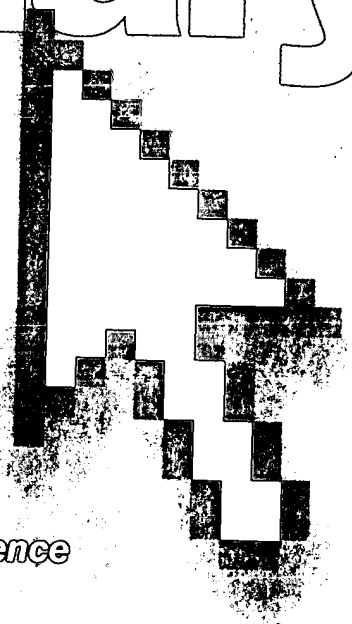
Microsoft® Press

Over 8,000 Entries

with online updates available quarterly

Microsoft® Computer Dictionary

Fourth Edition



- Three new appendixes, including Y2K, file extensions, and Internet domains
- Searchable text on CD-ROM
- Extensive coverage of hardware, software, the Internet, and more!
- Detailed illustrations and diagrams for easy reference

checking for errors. Most controllers require software that enables the computer to receive and process the data the controller makes available. *Also called* device controller, I/O controller.

input/output device *n.* A piece of hardware that can be used both for providing data to a computer and for receiving data from it, depending on the current situation. A disk drive is an example of an input/output device. Some devices, such as a keyboard or a mouse, can be used only for input and are thus called input (input-only) devices. Other devices, such as printers, can be used only for output and are thus called output (output-only) devices. Most devices require installation of software routines called device drivers to enable the computer to transmit and receive data to and from them.

input/output interface *n. See* input/output controller.

input/output port *n. See* port.

input/output processor *n.* Hardware designed to handle input and output operations to relieve the burden on the main processing unit. For example, a digital signal processor can perform time-intensive, complicated analysis and synthesis of sound patterns without CPU overhead. *See also* digital signal processor, front-end processor (definition 1).

input/output statement *n.* A program instruction that causes data to be transferred between memory and an input or output device.

input port *n. See* input/output port.

input stream *n.* A flow of information used in a program as a sequence of bytes that are associated with a particular task or destination. Input streams include series of characters read from the keyboard to memory and blocks of data read from disk files. *Compare* output stream.

inquiry *n.* A request for information. *See also* query.

INS *n. See* WINS.

insertion point *n.* A blinking vertical bar on the screen, such as in graphical user interfaces, that marks the location at which inserted text will appear. *See also* cursor (definition 1).

insertion sort *n.* A list-sorting algorithm that starts with a list that contains one item and builds an ever-larger sorted list by inserting the items to be sorted one at a time into their correct positions on that list. Insertion sorts are inefficient when used with arrays, because of constant shuffling of items, but are ideally suited for sorting linked lists. *See also* linked list, sort algorithm. *Compare* bubble sort, quicksort.

Insert key *n.* A key on the keyboard, labeled "Insert" or "Ins," whose usual function is to toggle a program's editing setting between an insert mode and an overwrite mode, although it may perform different functions in different applications. *Also called* Ins key.

insert mode *n.* A mode of operation in which a character typed into a document or at a command line pushes subsequent existing characters farther to the right on the screen rather than overwriting them. Insert mode is the opposite of overwrite mode, in which new characters replace subsequent existing characters. The key or key combination used to change from one mode to the other varies among programs, but the Insert key is most often used. *Compare* overwrite mode.

Ins key *n. See* Insert key.

install *vb.* To set in place and prepare for operation. Operating systems and application programs commonly include a disk-based installation, or setup, program that does most of the work of preparing the program to work with the computer, printer, and other devices. Often such a program can check for devices attached to the system, request the user to choose from sets of options, create a place for the program on the hard disk, and modify system startup files as necessary.

installable device driver *n.* A device driver that can be embedded within an operating system, usually in order to override an existing, less-functional service.

Installable File System Manager *n.* In Windows 9x, the part of the file system architecture responsible for arbitrating access to the different file system components. *Acronym:* IFS.

installation program *n.* A program whose function is to install another program, either on a storage medium or in memory. An installation program, also called a setup program, might be used to guide a user through the often complex process of setting up an application for a particular combination of machine, printer, and monitor.

Installer *n.* A program, provided with the Apple Macintosh operating system, that allows the user to install system upgrades and make bootable (system) disks.